# Certificate Management: A Practitioner's Perspective

Mike Whalen, Rockwell-Collins Inc.
mwwhalen@rockwellcollins.com

*Standards for critical avionics software development, such as DO178B, place a strong emphasis on process issues: ensuring traceability between different development artifacts and proper configuration management of these artifacts. Certification Management (CM) systems formalize many of the relationships between different artifacts and hold the promise of both streamlining the management of the artifacts and ensuring that relationships between the artifacts are formally justified. However, to be useful in an industrial context, the definition and scope of CM systems must be better understood, and several open issues must be addressed. This paper describes issues and potential uses of CM systems in industrial practice.*

## 1. Introduction

Current avionics software standards such as DO178B [6] focus on software development processes to try to ensure a high level of confidence in the correctness of the developed software. These process requirements include ensuring traceability between requirements, design artifacts, source, and object code, and also in maintaining proper configuration management between artifacts. However, little emphasis is placed on formal verification of functional behavior of systems.

Recent advances in modeling languages have made it feasible to formally specify and analyze the behavior of large system components. Synchronous data flow languages, such as Lustre [1], SCR [2], and RSML$^{-e}$ [3] seem to be particularly well suited to this task, and commercial tools such as SCADE [4] and Simulink [5] are growing in popularity among designers of safety critical systems, largely due to their ability to automatically generate code from models. At the same time, advances in formal analysis tools have made it practical to formally verify important properties of models to ensure that design defects are identified and corrected early in the lifecycle (see, for example, [8], [9], [10]). At Rockwell-Collins we are integrating formal analysis into the design and development cycle for next-generation commercial avionics systems and expect formal analysis to be an integral part of the V & V process for future systems.

Software certification management (CM) systems are designed to support independent verification of some aspect of software development. They introduce the notion of a *software certificate*, which contains all the information necessary for an independent assessment of the demonstrated properties. These certificates could be used to formalize many of the analyses that are required in guidelines such as DO178B, and also for formal functional verification of software artifacts, leading to safer systems.

Unfortunately, the current definition of CM systems is diffuse, and it is difficult to determine the boundaries between CM systems, configuration management systems such as CVS and Rational ClearCase, and requirements traceability systems such as DOORS. When managing informal artifacts, such as, for example, fault trees, textual requirements, or design rationales, the benefit of using a CM approach over traditional traceability tools is unknown. In order to use CM systems in an industrial setting, a more specific definition of the role and benefit of CM systems is required. This paper presents a few thoughts on how and where CM systems might be useful in a critical software development effort, and some future directions for research.

## 2. CM Opportunities for Showing Safety and Requirements Traceability in Critical Avionics Software

Critical software standards such as DO178B [6] require multiple levels and types of traceability between software artifacts. It distinguishes four abstraction layers of software artifacts: *high-level requirements, low-level requirements, source code,* and *object code*, and requires that the artifacts in each layer map to artifacts in the preceding and proceeding layer. The standard approach to satisfying DO178B uses a mixture of semi-formal analysis (often consisting of human inspections and checklists) and extensive testing to try to show that software is correctly implemented and corresponds to its requirements.

DO178B calls out several different kinds of analysis that should be performed on source code. Some of these analyses are designed to show conformance to higher-level requirements, while others are “well-formedness” checks to ensure that implementation does not allow safety or security violations. Many of the well-formedness criteria could be easily formulated as Proof-Carrying Code (PCC)-style safety policies to be proven of source or object code. These proofs could then be used as certificates for the system in question. Some well-formedness properties that are called out in DO178B are:

? unit-of-measurement/dimensional consistency between modules / subsystems

? arithmetic overflow/underflow

? variable initialization-before-use

? behavior of partial arithmetic operators (e.g. divide)

? termination

? deadlock/livelock/race conditions

? array/pointer safety

There are several tools that can automatically check such properties, such as PolySpace [13], but these currently do not generate evidence suitable for CM systems.

With the recent adoption of model-based development languages such as SCADE [1] and Simulink [5], it has also become easier to formally analyze functional behavior of software. These languages have relatively straightforward formal semantics that are straightforward to translate into model checking languages such as SMV [7]. Rockwell-

Collins has had significant success translating informal textual requirements into properties that can be proven against large software models [10].

In order to perform the proofs, it was necessary to split the software model into several *analysis models* and use techniques such as *temporal induction* [11] to perform assume-guarantee proofs. These creation of the analysis models and the *proof graphs* [11] were performed and justified by hand. In order to perform automated formal analysis of large systems, these kinds of steps will be necessary, and the hand-justifications are a weak link in the guarantees provided by the model checking tools. Certificates that ensure that the different analyses are correctly justified and related would be a significant benefit.

Another issue is that analyses performed by most model checking tools do not yield any kind of certificate, so cannot be justified. This leads to a situation where a significant amount of trust must be invested in the model checker. Another avenue for improvement would be the creation of certificate-generating automated analysis tools.

## 3. What is a Certificate Management System?

In order to use CM systems in avionics projects, we must better define what they are. According to the SoftCeMent web site, these CM systems look suspiciously like end-to-end CASE tools that require significant commitment of resources, including functions such as configuration management tools, databases, traceability tools, make tools, workflow tools, and audit and reporting tools. Businesses already have mature, established tools for most of these tasks, and it is unlikely that they will switch over to a single system for managing all of this functionality.

Also, there seems to be some fuzziness on what certificates are and how much assurance they can provide. Given a formal proof of some safety property on source or object code and the code itself, one can derive a very high level of confidence by mechanically checking the proof. On the other hand, given an informal safety property and an informally generated fault tree, what kind of guarantees can a certificate provide? In this case, it is difficult to see what benefit CM systems would provide over a simple code-signing approach provided by component tools such as Microsoft's .NET assemblies or Java JAR files.

We would suggest that most of the interesting and beneficial features of CM can be hosted as relatively self-contained "plug-ins" to existing tools. A tool like DOORS already has sophisticated traceability, linking, and reporting facilities. Plug-ins could be created to both help generate and check whether formal relationships hold between two artifacts within the database. Just this aspect of certificate management presents an enormous challenge, as there is a wide range of logics that can be used to check these properties and potential formats for certificates, not to mention challenges in automating the generation of certificates. It seems unnecessary and unwise to try to tackle software management issues that are capably handled by existing tools.

## 4. Conclusion

Formal tools and techniques are increasingly used in practice to help create and verify the behavior of critical software systems. Certification management systems have the potential to significantly streamline and formalize many of analyses that are required in avionics standards such as DO178B [6]. However, tools to support certification management are still in their infancy. To see significant industrial adoption, they must be fairly easy to use and integrate with existing configuration management and traceability tools. In order to be widely adopted, these tools must also integrate well into a certification story that is acceptable to authorities such as the FAA, since they ultimately decide whether a system is airworthy.

## References

[1] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R.de Simone, The Synchronous Languages 12 Years Later, Proceedings of the IEEE, Volume 91, Issue 1, January 2003.

[2] C. Heitmeyer, R. Jeffords., and B. Labaw, Automated Consistency Checking of Requirements Specification, ACM Transactions on Software Engineering and Methodology (TOSEM), 5(3):231-261, July 1996.

[3] J. Thompson, M. Heimdahl, and S. Miller.: Specification Based Prototyping for Embedded Systems, Proceedings of the Seventh ACM SIGSOFT Symposium on the Foundations on Software Engineering, LNCS 1687, September 1999.

[4] Esterel Technologies, http://www.esterel-technologies.com.

[5] James Dabney and Thomas Harmon, Mastering Simulink, Pearson Prentice Hall: Upper Saddle River, NJ, 2004.

[6] RTCA. Software Considerations In Airborne Systems and Equipment Certification (DO178B), RTCA, 1992

[7] IRST, http://nusmv.irst.itc.it/, The NuSMV Model Checker, Trento Italy

[8] Marco Bozzano, Antonella Cavallo, Massimo Cifaldi, Laura Valacca, and Adolfo Villafiorita, Improving Safety Assessment of Complex Systems : An Industrial Case Study. Proceedings of Formal Methods 2003 (LNCS 2805), Springer-Verlag, pages 208-222, 2003.

[9] R. Butler, S. Miller, J. Potts, and V. Carreno, A Formal Methods Approach to the Analysis of Mode Confusion, Proceedings of the 17th AIAA/IEEE Digital Avionics Systems Conference, Bellevue, WA, October 1998.

[10] S. P. Miller, M. P.E. Heimdahl, and A.C. Tribble, Proving the Shalls, Proceedings of FM 2003: the 12th International FME Symposium, Pisa, Italy, Sept. 8-14, 2003.

[11] K. L. McMillan, Circular Compositional Reasoning About Liveness, Cadence Berkeley Labs Technical Report 1999-02, Berkeley, CA, 1999.

[12] Telelogic, Inc. DOORS product description web page. http://www.telelogic.com/products/doorsers/index.cfm

[13] PolySpace Technologies, Inc. PolySpace product description web page. http://www.polyspace.com